

# DEFINIÇÃO DE CLASSES EM JAVA POR COMPOSIÇÃO

PROGRAMAÇÃO II

2015/16

©2016 LÍGIA FERREIRA, SALVADOR ABREU

# NA AULA PASSADA....

---

- Aprendemos:
  - ➔ a definir classes
  - ➔ reconhecer variáveis de instância
  - ➔ métodos de instância
  - ➔ construtores
    - Também:
      - tipo primitivo
      - declaração variáveis/afecção
      - instanciação
      - tipo de retorno
      - ....

```
class Nota{  
    int valor;  
    char moeda;  
  
    Nota(int n, char x){  
        valor=n;  
        moeda=x;  
    }  
  
    int valor(){  
        return valor;  
    }  
}
```



# CLASSES EM JAVA

- Uma classe com main

```
public class Teste{  
    public static void main(String[] args){  
        System.out.println("Olá!");  
        Nota x=new Nota(10,'€');  
        x=new Nota(1,'$');  
        System.out.println(x.valor());  
    }  
}
```

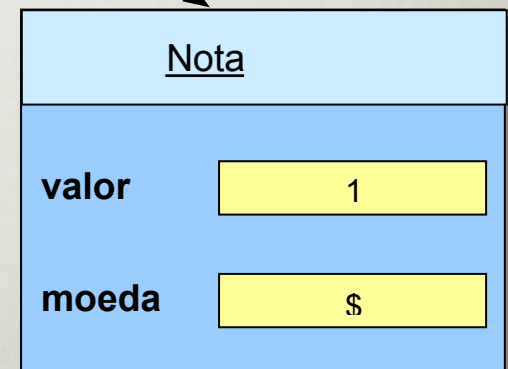
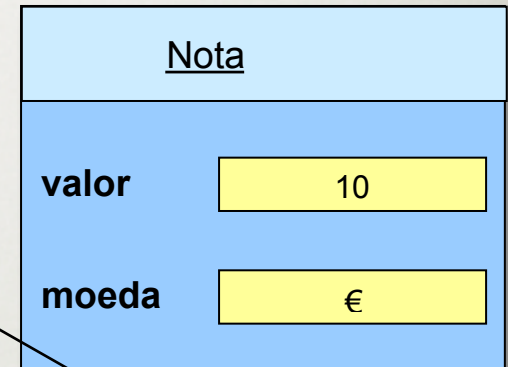
Olá!

1

x



**X**



# CLASSES E TIPOS

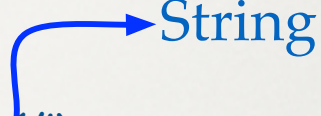
---

- As classes induzem um novo tipo de dados: os tipos referenciados
- Por cada classe que criamos temos um novo tipo, sobre o qual podemos
  - ➔ criar instâncias
  - ➔ enviar mensagens
- Não há necessidade de reinventar a roda, o java providencia muitos objectos que satisfazem as nossas necessidades. Devemos usá-los em vez de providenciar uma implementação que na grande maioria das vezes seria pior do que a que já existe.
- Devemos conhecer as classes e a sua API, para sabermos como usá-las
- Vamos abordar as Strings e a classe Math.



# STRINGS

---

- Já esbarramos com as Strings:  String
  - ➔ Quando fazemos `System.out.println("Ola")`
  - ➔ O método `System.out.println`, recebe uma instância de `String`
- Uma sequência de caracteres delimitados por aspas duplas são uma constante do tipo `String`
- Não existe o tipo primitivo `string`, mas uma classe `String`, que implementa as funcionalidades usuais para este tipo
- A classe `String` possui cerca de 50 métodos, não vamos falar de todos...
- Vamos também introduzir um operador para as Strings o `+`, o nome deste operador é `concatenação`

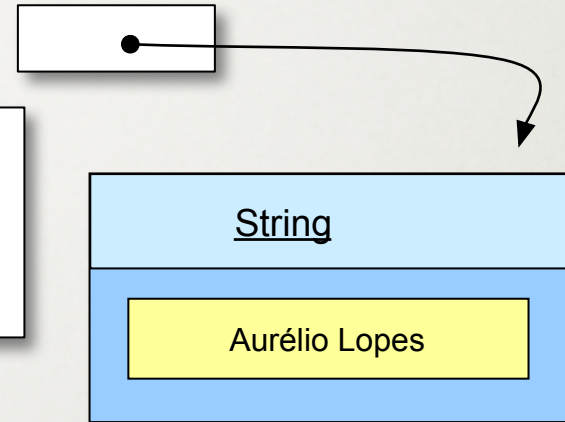
# STRINGS

1. O identificador nome é declarado sendo alocado espaço em memória para a variável

- As Strings são objectos

```
1 String nome;  
2 nome=new String("Aurélio Lopes");
```

nome



2. O objecto do tipo String é criado e o identificador nome referencia-o.

# STRINGS- OPERADOR +

---

```
String x="Era uma vez";  
String y="um gato maltês";  
String z= x+y;  
System.out.println(z);  
z=x+" "+y;  
System.out.println(z);  
System.out.println(x + "\n" + z);
```

```
Era uma vezum gato maltês  
Era uma vez um gato maltês  
Era uma vez  
Era uma vez um gato maltês
```



# CLASSE STRING

---

- Sendo as Strings implementadas como classe a inicialização
  - ➔ `String x=new String("Nova String")` é possível
  - ➔ Um dos construtores para a classe `String`, aceita como parâmetro uma sequência de caracteres delimitada por "", i.e uma `String`
  - ➔ Há outros, muitooooooooos até.
  - ➔ Alguns deles
    - construtor vazio: `new String();`
    - constructor que recebe um array de caracteres
    - `char[] palavra={'a', 'u', 'l', 'a'};`
    - `String s=new String(palavra);`

0	1	2	3
a	u	l	a



# FUNCIONALIDADES DA CLASSE STRING

MÉTODO	SEMÂNTICA
<p>serviços usuais para manipular <small>s</small></p> <p><small>char charAt(int i)</small></p> <p>int compareTo(String s)</p> <p>boolean endsWith(String s)</p> <p>boolean equals(String s)</p> <p>String String.valueOf(? a)</p> <p>int length()</p> <p>String substring(int i,int f)</p> <p>String toUpperCase()/toLowerCase</p> <p>String trim()</p>	<p>estes objectos. Eis alguns: <small>carácter no índice i</small></p> <p>-1, se &lt;, 0 se = , 2 se &gt;</p> <p>s é sufixo do receptor?</p> <p>o receptor e s objectos iguais?</p> <p>converte a para String</p> <p>comprimento</p> <p>substring entre i e f</p> <p>conversão Maiús(Minús)culas</p> <p>remove espaços (inicio e fim)</p>

# CLASSE STRING

---

- Não é possível, apresentar aqui todas as funcionalidades/métodos, disponibilizados pela classe String
- Quando for necessário uma funcionalidade que não conhecemos devemos:
  - ➔ Consultar a API da classe
    - Se tal funcionalidade existe, usa-se de acordo o método que a implementa
    - Senão, implementa-se a funcionalidade pretendida
    - Resumindo: Não inventar a roda.



# EXEMPLOS

---

```
String x= new String("Era uma vez");  
System.out.println(x.charAt(2));  
System.out.println("Ana".compareTo("Pedro"));  
System.out.println(x.endsWith("s"));  
System.out.println(String.valueOf(1999));  
System.out.println(x.length());  
System.out.println(x.substring(8,11));
```

```
a  
-15  
false  
1999  
11  
vez
```

# MÉTODOS ESPECIAIS

- É usual definir nas classes uma representação em String dos objectos criados.
- Qualquer método que devolva uma String é passível de ser usado, mas usemos um "standard"
- Método público que retorna uma String e que se chama toString, sem parâmetros

Já tínhamos "visto" este modificador

modificador	retorno	nome
public	String	toString()

```
/*código que define  
a representação em String do objecto */  
return essaString; }
```

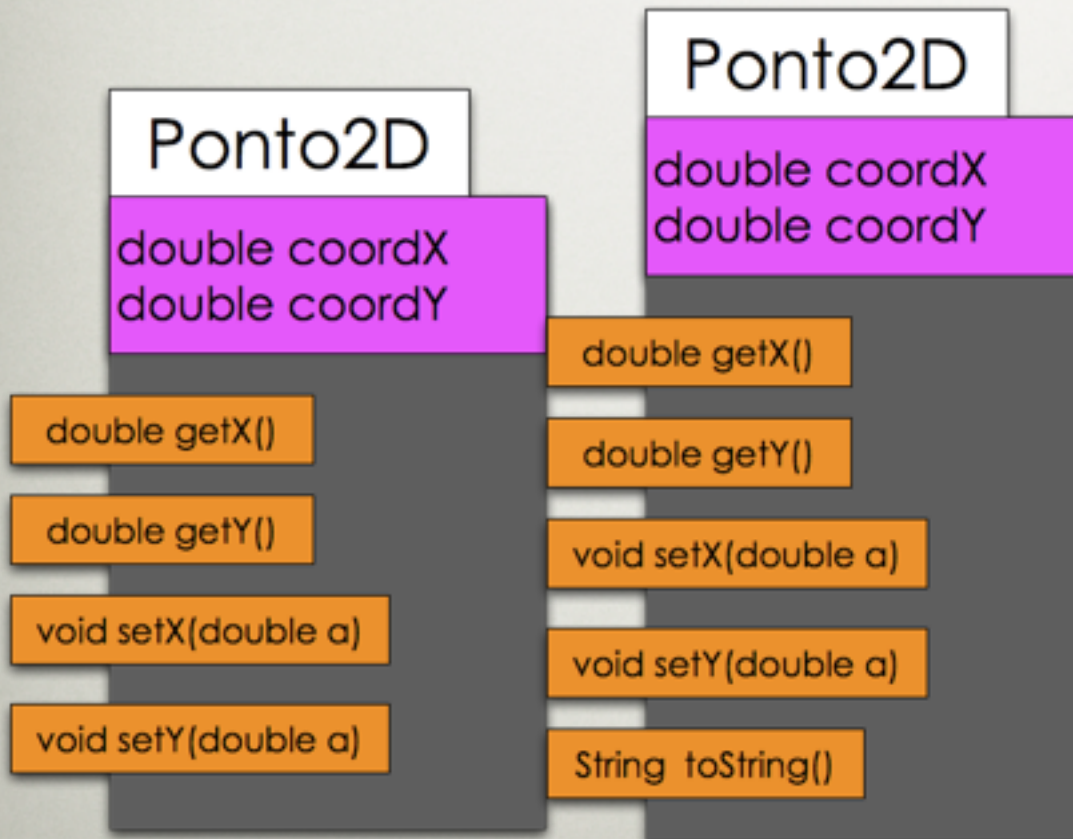
modificador	modificador	retorno	nome	declaração	parâmetros
public	static	void	main	(	String[] args)

```
{  
....  
}
```



# MÉTODOS ESPECIAIS

- Exemplo de toString para pontos2D

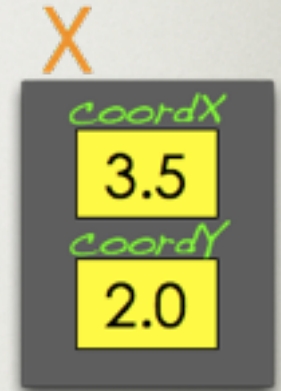


```
public String toString(){  
    String s=new String("(" +coordX+  
        ","+  
        coordY+  
        ")");  
  
    return s;  
}
```

# MÉTODO toString()

```
Ponto2D x=new Ponto2D(3.5,2);  
String s=x.toString();  
System.out.println(x.toString());  
System.out.println(s);  
System.out.println(x);
```

(3.5,2)  
(3.5,2)  
(3.5,2)



SE

é esperada uma String  
e existir na classe um método de assinatura **public String toString()**

ENTÃO

**toString** é invocado automaticamente para converter em String o  
objecto



# CLASS MATH

---

- A classe Math é uma classe “especial”, dado que não é permitido criar instâncias da classe, i.e. não é possível fazer `Math m=new Math()`
- É uma classe de grande utilidade dado que disponibiliza um conjunto vasto de funcionalidades na área da matemática
- Dado que não podemos criar instâncias da classe, para usarmos as funcionalidades da classe, enviamos mensagens, não às instâncias mas à classe
- Exemplos:
  - ➔ `Math.sqrt(5.32);` // raiz quadrada de 5.32
  - ➔ `Math.cos(7.54)`
  - ➔ `Math.PI` // constante com o valor de pi
  - ➔ `Math.pow(2,8)` //  $2^8$
- A seu tempo veremos como podemos que as classe criadas por nós, possam fazer algo semelhante...

# RANDOM

---

- Random significa aleatório
- A classe Math() disponibiliza um método random()
- Consultando a api da classe vemos que random:
  - ➔ "Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0."
  - ➔ i.e.  $0 \leq \text{Math.random()} < 1$
  - ➔  $0 \leq \text{Math.random()} * N < N$
  - ➔  $P \leq \text{Math.random()} * n + P < N + P$
  - ➔ Temos aqui uma forma para gerar números aleatórios
- Existe também a classe Random
  - ➔ Obviamente também permite gerar números aleatórios